Architecture Knowledge Management during System Evolution – Observations from Practitioners

Ipek Ozkaya
SEI/CMU
4500 Fifth Ave
Pittsburgh, PA, USA, 15221
+1-412-268-3551
ozkaya@sei.cmu.edu

Peter Wallin
Mälardalen University
P.O Box 883
SE-721 23, Västerås, Sweden
+46 21 103198
peter.wallin@mdh.se

Jakob Axelsson
Mälardalen University
P.O Box 883
SE-721 23, Västerås, Sweden
+46 31 597003
jakob.axelsson@mdh.se

ABSTRACT
It is widely accepted that awareness of architectural decisions enables better management and planning of system evolution, refactoring, and modernization efforts. In this paper we report data from interviews with software architects about how practitioners utilize architecture during system evolution. Our results show, despite the widely shared view that long-lived systems are better off with strong architectures; basic architecture-centric practices are not followed systematically. The key gap we observe is in correct and timely communication of architectural issues. This overall finding is not surprising. However, our data also contributes to how architecture knowledge management activities can be focused for most benefit throughout a system's lifespan. While the often-referenced problem is lack of time spent on documentation and design practices, our interviews show that lack of quality attribute reasoning early on, and during the lifespan of the system is a key contributor to failing to use architecture knowledge effectively during evolution.

Categories and Subject Descriptors
D.2.11 [**Software Engineering**] Software Architectures

General Terms
Management, Design

Keywords
Software architecture, system evolution, architecture-centric practices, architecture knowledge management

1. INTRODUCTION
The creation and adaptation of software has significantly shifted to a model where existing systems evolve to meet changing business needs. There is evidence in literature that supports the importance of architecture in managing system evolution [5] [6] [11] [15] [18] [19]. The architectural knowledge (AK) management research community has also emphasized one of the key benefits of AK management as supporting system maintenance and evolution [7] [12] [13].

Understanding the best form of support for system evolution is challenging, as observing evolution projects requires immersing within the project for an extended period of time. Issues often do not come from one source, or can be resolved with one particular technique, discipline, or strategy alone. While there has been significant amount of theoretical work in techniques and tool support for evolution, understanding their use in practice when it comes to architecture-centric practices has not gained much empirical attention.

In order to understand the state of practice in utilizing architecture-centric practices and knowledge for system evolution, we pose the following questions:

1. *Are architecture-centric practices used as means to guide evolution systematically?*
2. *Which practices are used?*
3. *Which practices are systematically omitted?*

As our method of investigation, we designed a survey outlined as a structured interview, conducted one-on-one. The interview consisted of thirty seven open-ended questions – questions that required descriptive answers – targeted towards system and software architects. The questions spanned from the use of key architecture-centric practices in general to how they were focused in addressing evolution issues. We present results from nine architects working in domains ranging from defense, telecommunications, automotive, healthcare, and manufacturing automation.

The key issues in use of architecture-centric practices to guide evolution were common across our interviews. When it comes to AK management, in contrary to the common perception that industry views architecture documentation and time spent in design as possible overhead time, we found that these practices were among the most common practices. What is revealing from an AK management perspective is the lack of attention given to eliciting and utilizing architecturally significant requirements throughout the systems lifespan, and a lack of focus on quality-based reasoning.

The problem remains to be the mismatch between the needs of projects under market, business, economic, and customer constraints, and the lack of techniques for capturing critical decisions to impact healthy evolution of systems. The collected data provides empirical support that systematical use of architecture-centric practices do not serve as a first class resource when it comes to collecting and acting upon knowledge during system evolution in these large-scale projects.

In this paper, we present data and observations from the interviews conducted. We aim to draw attention to the gap in use of architecture-centric practices as prescribed versus as practiced even though architecture knowledge remains to be critically sought for. The rest of the paper is organized as follows. In Section 2, we survey related work. We describe the followed empirical method in Section 3. In Section 4 we present our results. Section 5 presents key observations and concludes the paper.

2. SYSTEM EVOLUTION AND AK MANAGEMENT

Successful system evolution means ensuring present and future alignment of the system to business and mission goals in a manner to maximize value and reduce risk. Architecture-centric practices enable a product's definition, development, and evolution.

Existing research on architecture evolution focuses on the use of complexity metrics to measure architecture evolvability [3] [9]; planning, generation and analysis of architectural evolution paths [17] [11]; application of economic analysis [23] [3]; and focusing evaluations to manage evolution, such as within a product line [8].

Empirical work on software evolution has made significant contributions in improving our understanding of system change and degradation on long-lived systems [6] [15] [18]. Yet, evidence on the effectiveness of evolution techniques, tools, and methods on practice has been sparse at best [16]. A case study at an automotive OEM indicates drivers, trade-off areas, and technical solutions related to evolutionary architecting [2]. A recent questionnaire-based study on risks in software architecture evolution, report that among IT-professionals they are mostly observed during planning [22].

In the context of IT application and architecture design, the concept of "transitional architectures" have been suggested to realize system architecture on a time continuum [10]. Conceptually, the approach is based on three activities: i) understand the current state of the architecture, ii) envision a desired future vision of the architecture, and iii) establish a sequence of discrete steps by following a gap analysis process between the current and the envisioned architecture. Such an approach suggests the creation of an evolution path based on the envisioned architectural changes.

Another class of work that focuses on the evolution path notion aims to formalize common evolutions as architecture evolution styles. Garlan [11] defines an evolution style as a set of evolution paths among classes of systems, e.g., evolutions from a web-based architecture to J2EE. Le Goaer et al. [12] define an evolution style at a much lower level of abstraction in terms of the structural changes involved.

Architectural knowledge management and design decision support has recently gained an increased attention in the research community. Architectural knowledge is defined as the integrated representation of the software architecture of a software-intensive system or family of systems along with architectural decisions and their rationale external influence and development environment [1]. A key immediate benefit that motivates work in AK management is the potential support AK can provide during evolution activities [7] [12] [13].

Work to date in AK management has two leading threads: design decision modeling and ontological views of AK, and tool support for visualization and management of AK. Capilla et al. present a meta-model that integrates project, architecting, and decision models [7]. It is possible to view the architecting aspect of this model as a way of capturing the practice-centric view of the architecting process. Kruchten uses the ontology view to organize different types of decisions and their attributes [14]. Kruchten [13] and Babar [4] summarize the framework and tool focused approaches. Babar highlights that all of the existing efforts focus on codification of existing knowledge for the goal of storing and searching, as opposed to personalization that focuses on helping people communicate knowledge.

3. EMPRICAL SURVEY METHOD

In this section, we describe the research method used to investigate the questions presented in Section 1.

As our primary approach to collect data, we used structured interviews with open-ended questions [20]. Structured interviews can be seen as a guided survey that is conducted face to face or over telephone. The major drawback of this approach is the relatively time consuming procedure to perform one-on-one structured interviews. This reduces the sample size due to the effort it takes to conduct such interviews. However, there are several advantages as well [24]:

- One-on-one interviews ensure a high response rate.
- They ensure collecting data for all the questions and reducing ambiguities.
- The respondents have the opportunity to ask clarification questions when needed; hence the risk of misinterpreting the questions is reduced.
- The interviewers can encourage the respondents to elaborate their answer further.

3.1 Context

The subject of analysis for this study is system and software architects, describing their use of architecture-centric practices on a recent or current project. The participants were from a wide range of companies located in Europe and North America. Common to all of the projects and companies is that they develop software or software-intensive products.

In Table 1, we present the demographics of the nine projects we collected data for. The data is categorized based on the domain from which the respondents reported their experiences. We have respondents from telecom (T), automotive (A), health care and health insurance (H), defense contractor (D), and manufacturing automation (M). In order to provide insight to the range of organization, we also report the companies' footprint. We use the companies' ranking on the fortune global list of the largest companies where applicable, otherwise their number of employees.

Each respondent focused on a current or recent software development project, in which evolution was a concern. In order to provide the context for each project, respondents were asked to describe their product according to the product's place in the market, development team size, and size of system. Respondents used different metrics to categorize the size of their systems. Since the type of metric give insights about how projects are managed and planned, we did not enforce one particular metric to be reported. Instead we asked the respondents to give the key measurement that describes the size of the project within their organizations' context. Development team size, expected development cost, number of classes, size of documentation, and expected source lines of code were among the metrics used by the respondents for describing project size. All of the products that were subject of discussion had 10 to 20 years of life expectancy.

Table 1: Summary of participants

Respondent	Company footprint	Place in market	Project team size	Size of system
T-1	Fortune Global 1000	Top 2	40	Unpredictably large. No metrics used.
T-2	Fortune Global 500	Market leader	150	6 volumes of architecture documents - smallest 100 pg.
T-3	Fortune Global 500	Market leader	60	1-300 processors
A-1	Fortune Global 500	Strong follower	150	1 million SLOC
A-2	Fortune Global 500	Follower	100	Distributed system with multiple control units
H-1	Global ~7500 employees	Market leader	15	$1 million
H-2	North America ~1000 employees	Market leader	50	100 KSLOC (In house developed)
D-1	Fortune Global 500	Market leader	80	1,3 million SLOC total, 580 KSLOC in-house
M-1	Fortune Global 500	Follower	5	5 people full time in 3 years

3.2 Planning and Preparation

We interviewed chief or lead architects. All the respondents had a long background in designing software intensive systems, most respondents with more than 15 years of experience. It should be noted that we report the experiences and views of the interviewed people and not necessarily the general trend at that particular business unit or company. However, since many questions relate to whether a certain practice is used or not it is likely that the respondent is representative for the organization.

We prepared the questions in two phases. Initially we generated a list of key questions – this list had 103 initial questions. Since our goal was to make this a structured interview that busy professionals were willing to participate in, we went through an exercise of prioritizing the questions. We concluded that our purpose would be covered with 37 questions.

3.3 Interviews

All interviews were conducted using the same set of predefined questions. The order of the questions was consistent for all interviews. The duration of each interview ranged between 90 to 120 minutes.

No recording devices were used to further ensure that the respondent spoke as candidly as possible. Two researchers were present at all interviews; both took notes, and asked clarification questions. Since all interviews except three were made as a conference call, the questions were sent to the respondents beforehand to make the interview process easier. After the interview was concluded, the notes from the two researchers were merged, and respondent clarification and approval was collected were applicable.

The questions were divided into five different categories: experience; general company and project information; use of architecture-centric practices; evolution triggers; and evolution and architecture-centric practices.

3.3.1 Experience

This first category served the purpose to learn about the experience of the respondent, years with the company, current and past positions, and experience with software architecting. Since the interview in many cases was the first contact the respondent had with the interviewers, the introductory questions also served as a way to get the respondent comfortable and eased into the interview.

3.3.2 Company and project information

Company and project information questions aimed to collect details about the projects that the respondents reported experiences from. Examples of questions in this section are the following.

- *What is the place of the business unit/company in the market?*
- *What is the size of system and software development unit for the project? (SLOC, number of classes, number of developers, and/or other descriptive metrics).*

In addition, information about the contextual constraints around the system such as legacy-dependency or product-line environment was elicited as part of general information. This section also included questions about project time lines such as life expectancy of the product and project duration.

3.3.3 Use of architecture-centric practices

The motivation for this section was to get a basic understanding of the architecture-centric practices used in the particular project referred to by the respondents. Discussions were around basic practices starting with how the project elicited architectural significant requirements, and if they were documented separately from the functional requirements. In addition, data about how the architecture was designed and analyzed was collected, together with questions about architecture documentation, evaluation and how they ensure conformance between the documentation, implementation and the work product. Some example questions are the following:

- *Do you evaluate architectural designs, and if so, how?*
- *Is the documentation up to date with the architecture and the system?*
- *Who uses the architecture documentation?*
- *Is there traceability from the architectural model to work products and how are they kept synchronized?*

3.3.4 Triggers for evolution

The triggers for evolution concerns questions that relate to the causes of system change and how these causes affect the architecture and the system. Questions aimed to elicit data about both expected and unexpected triggers for evolution. The questions in this section were aimed at system evolution to understand how the discussed project worked with such. Example questions from this section of the interviews are the following:

- *What are the primary causes for evolution/system change?*
- *How do you handle unexpected evolutions, such as market change, technology change, and domain change?*
- *Are there commonalities among different evolution cycles?*

3.3.5 Evolution and architecture-centric practices

In the last section, our questions were about how they deal with evolution issues that the respondents expressed concerns about in the previous section.

- *Is architecture used as a basis for evolution?*
- *How do you plan for evolution?*
- *How do you plan and communicate the architecture for different timelines, 6 months, 1 year, 5 years?*
- *How often do you need to change the architecture?*

Such questions had the purpose of collecting data about the use of architecture-centric practices before, during, or after key evolution triggers were met within the project.

3.4 Data Analysis

All data were stored in a spreadsheet. A chain of evidence was upheld by a case study database as described by Yin [25]. All data analysis was done by two researchers together enabling discussion about how to interpret the data.

We used statements from the different respondents to get a consistent view of how projects handle each of the different topics from the study. Two of the questions asked were ranking questions. The results from these questions are presented as frequency tables. In addition, many questions resulted in key categories to emerge within the results, we also present this data by frequency within the categories. In the correlation analysis, we tried to find correlations between different questions, i.e. is there a relation between certain architectural practices and if architecture is used to guide evolution; however, based on the small data set statistical correlations are not possible to make.

4. RESULTS

We present the results divided into two major categories: use of architecture-centric practices in general, and use of architecture-centric practices for evolution.

4.1 State of practice

In Table 2 we summarize the usage of architecture-centric practices the interviewees talked about. We used three categories to differentiate to what extent the practices were used: *standard practice, ad-hoc practice* and *did not use practice*. We define standard practice as a practice carried out in a structured and systematic way. Ad-hoc refers to practices that are followed in an unsystematic manner and usually not part of the companies' process. If a practice was not used at all, it was categorized as did not use practice.

4.1.1 Architecturally significant requirements

Producing high quality architecture is closely dependent on understanding architecturally significant requirements [4]. Architecturally significant requirements have direct impact on the design decisions and tradeoffs made. Methods and tools that are used for eliciting, documenting, and managing architecturally significant requirements, such as quality attribute scenarios, therefore are among the key architecture-centric practice areas.

Table 2. Usage of architectural practices

Architectural practices	Standard practice	Ad-hoc practices	Did not use practice
Documentation	6	3	0
Evaluation	3	6	0
Reconstruction	0	5	4
Explicit design	7	2	0
Architectural requirements	2	7	0
Elicitation of business goals	5	4	0

Noteworthy is that many of the respondents did design an architecture and document it, but at the same time almost no one elicited and document architectural significant requirements in an explicit way. Seven out of the nine participants replied without hesitation that architecturally significant requirements were only intuitively known based on the experience of the architects and developers. The notion of architectural significant requirements we addressed is not a high-level notion of quality, such as modifiability, performance, but a specific understanding of measurable quality drivers beyond functionality. Time to explicitly elicit architecturally significant requirements was not allocated. They were extracted from the functional and user requirements in bits and pieces, but not as key architectural drivers. The remaining who did allocated time for architecturally significant requirements followed different practices. In one case the IEEE 830-1998 requirement specification standard was followed to provide a taxonomy. Out of the nine, only one respondent described existence of multiple architect roles within the organization ranging from enterprise architect, system architect, and to application architect. In this case architecturally significant requirements needed to be approved by the enterprise architect; hence explicit practices in their elicitation and management existed.

In response to the question how often requirements change and how often architecturally significant requirements change, five of the participants said that requirements change almost daily and the remaining said that requirements do change, but did not have a significant impact on their tasks. The more interesting observation was on how architecturally significant requirements changed. While requirements changed for half of our participants, all of our participants said that architecturally significant requirements do not change often. Key architectural drivers are there to stay, and when a change does happen it reflects a significant issue.

The results indicate that the notion of architecturally significant requirements was viewed at a very high-level. As many of our respondent did not have explicit practices to elicit and manage them, it is also quite possible that key changes went by unnoticed. Even if changes happened, these requirements did not change, and this knowledge was thus not as critical. However, we also observe a communication gap. Although all of our participants were aware of notion of architecturally significant requirements and methods to specifically share the knowledge about them, such as quality attribute scenarios, they did not use these techniques, resulting in loss of information and misconceptions that these requirements did not change in the global sense.

4.1.2 Design and Analysis

The areas we focus on for understanding architecture-centric design and analysis practices are design of target architecture, prototyping, reconstruction, conformance, evaluation, and documentation.

We explicitly asked the question whether the target architecture was designed and documented. Half of our participants said that this was done as an up-front effort. The remaining expressed that architecture was at a very high-level and details emerged throughout the iterations. However, eight out of nine participants used prototyping as part of their architecting process, especially when they needed to understand hardware to software issues prototyping became a major design technique. Out of these eight, seven used prototyping as exploratory and throw-away prototypes, in the last case there was mixed use of throw-away and evolutionary prototyping. Prototyping was conducted with ad-hoc practices; knowledge gained during the process was not captured.

Tool-based architecture reconstruction techniques were unanimously not used. Seven out of nine participants; however, indicated that evolving legacy software and dependence on legacy code was a major constraint in the project they reported about. All of the participants ensured conformance of architecture with implementation in an ad-hoc manner. They resorted to active communication by ensuring the participation of the architect within the development efforts, a clear need where AK were sought for through ad-hoc methods. One respondent's description of their reconstruction technique was representative of how hard a time architects have when they need to motivate the need for added work on architecture to collect knowledge.

> *"The management is more willing to call people back from retirement to deal with the issues of the legacy systems than having us spend time on understanding and reconstructing their architecture. It is not easy to have them understand we may not be as lucky next time."*

Similarly, architecture evaluations were informally done in the majority of the cases: in three projects formal evaluations were used. Two reported using SEI ATAM and one AT&T's questionnaire technique [5]. In two instances, evaluations were conducted as part of a gated project management process where project managers or senior engineers needed to approve the design. In these cases the goal was not to compare different design alternatives, but to accept the current solution as part of milestone approval. The remaining four reported not using any evaluation technique apart from need-based local technical reviews with ad-hoc means to capture the results.

Four respondents felt comfortable that the documentation was up-to date with the actual system being deployed. Out of these four; however, one said it was up-to date because what was referred to as architecture was very high-level. The other three said updating the architecture was an enforced requirement of their project management and customer deliverables requirements. All of these respondents said that the documentation was used by team members as a source of architecture knowledge.

What we observe from these responses is an inconsistent state of practice when it comes to what key architecture-centric practices are utilized. In many cases we also observed that architecture was thought of as the high-level contextual picture and all the rest of the decisions were grouped within implementation and detailed design tasks, creating a potential knowledge gap where critical information about elements and their relationships gets unrecorded.

4.2 Evolution and architecture

We investigated evolution under two categories from the perspective of architects: triggers of evolution and use of architecture-centric practices to manage evolution.

Table 3. Primary triggers for evolution

Triggers for evolution	# of responses
New features	4
Market change	2
Technology obsolescence	2
Scalability	2

4.2.1 Causes for evolution

Table 3 shows the evolution trigger responses. To get a list of triggers we asked the respondents to give the primary causes for system change. We did not provide them with a predetermined list. The repeated mentioning of scalability is noteworthy. For all the respondents scalability referred to the ability to add resources to the infrastructure with ease in order to support a larger customer base or technology change. Maintainability; hardware changes; change of requirements due to change of business drivers; commonality requirement among business units; were also mentioned as triggers for evolution but only mentioned by one respondent and therefore not included in Table 3.

We also asked whether such triggers arrived as unexpected changes and how they were handled. All of the respondents said that the evolution triggers were often expected or did leave enough time to plan for them. Even those respondents that called out scalability as a key issue said that to architect for a less scalable architecture was a decision taken to manage time to market requirements. They took the route to re-architect the system during a future release if needed, being aware of the cost impact of re-architecting.

4.2.2 Evolution and architecture

Given our insights in architecture-centric practices, we focused on evolution planning, economic trade-off analysis and using architecture for evolution. We initially focused on architecturally significant requirements that could be significant for evolution. We asked the participants to rank architecturally significant requirements that have been previously determined to be significant within the evolution context [19]. The respondents were given a list of requirements and asked to rank them Low (L), Medium (M), or High (H) respectively. In order to avoid semantic mismatches we provided the participants with definitions for these key quality attributes. Note that these may not be the most important quality attributes for the project, but they were ranked in comparison to each other. Stability and maintainability were the key requirements that were ranked as high by most participants as shown in Table 4.

Table 4. Architecturally significant requirements for evolution

Architecture significant requirements	L	M	H
Stability	1	2	6
Maintenance	1	3	5
Flexibility	2	3	4
Extensibility	2	4	3
Modifiability	2	4	3
Reusability	3	3	3
Evolvability	2	5	2

All of the respondents worked with 12-18 month major external release cycles. Typically within a 12 month period, they aimed for one major and 2-4 minor releases. In addition, for all of the respondents, planning at the level of architecture was conducted only for the current major external release cycle. While some high level decisions were known, very little was done within the current release cycle to support a long term plan. Planning around architecture was best described by one of our respondents as:

> *"When I am planning ahead 6 months, I am being very strategic."*

None of the respondents used architecture-level quantitative cost-benefit analysis. All of them mentioned that if schedule and budget slips were an issue, resources were cut from architecting, and they had little to no chances of obtaining funding for architecture-level projects where obviously observable one-to-one mapping with current feature needs did not exist. In essence, this ruled out activities, such as refactoring, reconstruction, or evaluation creating major knowledge gaps. One architect gave us interesting insights about the impression of value, cost, and architecture at the management level.

> *"Many of our evolutionary changes could not be labeled as architecture changes because the moment you tag a change as architectural the connotations it ad in the organization was that it would be costly, it would be time-consuming, and it would still be late to market. This created major issues in the project as the problems could never be addressed adequately."*

The bottom line of what we heard is that architecture knowledge is important, but practitioners do not utilize architecture-centric practices consistently in regular project life cycles to manage key architectural decisions, let alone when evolution challenges hit. And surprisingly, the main gap was not in foregoing documentation or architecting – which was our assumption going in – but in spending time with understanding architecturally significant requirements both during the initial phases of projects and when evolution issues hit.

4.3 Validity

An important aspect of interview studies is to ensure that both the method and conclusions made from the result are valid [24] [25]. Although the sample size is fairly small, we still argue that it is large enough for our conclusions. We base this on five key elements:

- *Broad span of domain, and international markets.* We had respondents from seven different key domains that represent high market share. Also, most companies that the projects were drawn from are multi-national and are considerably important players within their domain.
- *Evolution scope of projects.* All the projects had similar long life expectancy with both evolution and maintenance concerns where architecture knowledge is needed.
- *Broad span of complexity of the projects.* As shown in Table 1, the size and complexity of projects ranged in various dimensions.
- *Respondents experience.* All respondents had extensive experience in system and software architecture development. All interviewees had also been with that particular company for a long time and should be well aware of the practices used to apply in their current context.
- *Theoretical saturation.* After a number of interviews the responses started to get repetitive and towards the end of the series close to no differentiating answers were collected. As explained in [21], when responses start repeating any new data would only add, in a minor way, to the many variations of major patterns. Our sample size provided us data to observe key patterns to address our initial questions.

5. DISCUSSION AND CONCLUSION

Despite the extensive research and theory and strong impression of professionals that architectural thinking is important in defining, developing, and evolving large-scale long-lived systems, architecture-centric practices are still not systematically followed. Yet, AK continues to be important, team members strive to convince management for resources to spend more time in activities to obtain AK. There is clearly a mismatch. When resource constraints hit, architecture-centric practices are the first to be omitted from project planning, resulting in key decisions to be communicated ad hoc.

In summary, based on this study we answer our initial questions as the following.

1. *Are architecture-centric practices used as means to guide evolution systematically?*

Architecture-centric practices are not followed systematically. Within the projects we surveyed, architectural success of products relies heavily on the experience of the architects assigned to projects, rather than on the outcomes of systematic practice. When such is the case, AK is managed in an ad hoc manner. Research-based architecture tools and methods, and common mature architecture-centric practices are known about, but are not used.

2. *Which practices are used?*

The architecture-centric practices that are most commonly used are high-level architectural design and documentation. Documentation is used advantageously as needed. The documents may be out of sync with the system. Prototyping appears to be a key practice used to focus design. Our results did not converge on common techniques across different projects for these practice areas.

While this is encouraging, what architecture means varies significantly in terms of the level of detail expected from architectural decisions. Hence, often such documents are useless for carrying AK.

3. *Which practices are systematically omitted?*

While extensive requirement elicitation and representation practices exist, architecturally significant requirements are not elicited and managed explicitly. Such management is left to the experience of the architects. Similarly, architecture-level planning is conducted at best with ad hoc approaches and is not utilized commonly to manage evolution.

In this study, we took the view that since architecture as an artifact describes the structure of a system, a key aspect of a system's lifespan where architecture and AK would be essential is during evolution. Therefore, practices that are used for creating and evolving systems are essential in eliciting, communicating, and using AK.

Our observations from these interviews are still convincing that architecture-based system evolution is critical.

- Systems are expected to be in service and maintained for extended periods of time.
- Architecture stability and scalability to meet future needs are key drivers for organizations.
- Projects will continue to depend on existing products and evolve from them.

Even though all the respondents emphasized architecture work as an important success factor for their products, they still seem to need to justify the time spent on architecting and associated architecture-centric practices. Consequently, AK knowledge is managed ad hoc or companies rely heavily on experience.

Based on the result in Section 4, we observe:

AK to support effective evolution is not explicitly managed. Six of the nine respondents had uncertainty of market among their top three uncertainties. Also, the main triggers for evolution are external factors such as new features, market changes, and technology obsolescence. This could indicate that putting a lot of effort into the architecture evolution before needed is not top priority. On the other hand, the majority of key architectural concerns were to achieve stability, which requires AK about what is achieved and what is put-off to be explicitly recorded.

- *There is a high reliance on experience when it comes to architecting and AK management.* Experience and following practices were almost at odds in our interviews. The more experienced people the less they put emphasis on the processes and practices, yet they were able to put successful systems out of the door. One of our respondents even said, "If you were not around when the technology emerged in the market and followed how it evolved you can never be an architect for this kind of a system."

- *Architecturally significant decisions are not recognized as architecturally significant during evolution.* Many architecturally significant decisions are either postponed to development time or completely omitted. This is evidenced by the fact that key requirement changes do not change architectural concerns and change does not affect the document since the documents are just high-level. It is also evident by the reluctance to call out architectural change in a fear that it will be tagged too costly and timely.

- *There is no success criteria explicitly related to architecture, hence reducing the perceived importance and value of AK.* Revenue is the leading project success criteria for all the products in review. However, none tracked project success and revenue through the architecture. On the other hand, architectural change as all of our respondents alluded to is something that they want to stay away from because it is costly and hard to get acceptance for. The mismatch between success and architecture requirements is clearly one of the underlying causes for neglecting architecture practices.

Arguing that practitioners are to blame for not systematically adopting architecture-centric practices is clearly not the issue here. Neither is the issue going back to the drawing board and tweaking existing practices to increase adoption rate. In this regard, the results of our interviews are not surprising at all. What is evident; however, is despite the lack of systematically following architectural practices, AK is created and used, large systems are in service for long periods of time, even if with suboptimal use of resources to support system evolution. Yet, practitioners still do not have the vocabulary to talk about and motivate key architectural decisions and their impact to their management. The results of our study emphasize the need to focus methods, tools, and practices from the perspective of their support for AK management and decision support perspective that is critical for evolving systems. This, we believe will also impact transitioning more architecture-centric practices to routine software engineering to improve how systems are created and evolved.

6. ACKNOWLEDGMENTS

We thank the participants to our interviews. We also thank Len Bass from the Software Engineering Institute, Stig Larsson from the ABB Group, and anonymous reviewers for their valuable comments.

7. REFERENCES

[1] Avgeriou, P. et al., 2007. Sharing and Reusing Architectural Knowledge--Architecture, Rationale, and Design Intent. In Companion to the proceedings of the 29th International Conference on Software Engineering. IEEE Computer Society, 109-110.

[2] Axelsson, J. 2009. Evolutionary Architecting of Embedded Automotive Product Lines: An Industrial Case Study. In Proc. WICSA 2009.

[3] Bahsoon, R., Emmerich, W., and Macke, J. 2005. Using Real Options to Select Stable Middleware-Induced Software Architectures. IEE Proc. Software, 152(4).

[4] Babar, M.A. and Gorton, I. 2007. Architecture Knowledge Management: Challenges, Approaches, and Tools. In: Proceedings of the 29th IEEE International Conference on Software Engineering. (ICSE07) Companion, May 20-26.

[5] Bass, L., Clements, P., and Kazman, R. 2003. Software Architecture in Practice, Second Edition. Boston, MA: Addison-Wesley.

[6] Belady L., Lehman M.M. 1976. A Model of Large Program Development. IBM Systems Journal 15 (1), 225 -252.

[7] Capilla, R., Nava, F. & Duenas, J.C., 2007. Modeling and Documenting the Evolution of Architectural Design Decisions. In Proceedings of the Second Workshop on Sharing and Reusing architectural Knowledge Architecture, Rationale, and Design Intent. IEEE Computer Society, 9.

[8] Del Rosso, C. 2006. Continuous evolution through software architecture evaluation: a case study. J. of Software Maintenance: Research and Practice, vol. 18. 351-383.

[9] Eden, A. H., and Mens, T. 2006. Measuring Software Flexibility, IEEE Software, 153(3), 113–126.

[10] Erder, M. and Pureur, P. P. 2006. Transitional Architectures for Enterprise Evolution. IT Professional, May/Jun (2006) 8(3), 10-17.

[11] Garlan, D., M. Barnes, J., Schmerl, B., and Celiku, O. 2009. Evolution Styles: Foundations and Tool Support for Software Architecture Evolution. In Proceedings of WICSA 2009.

[12] Goaer, L., Tamzalit, O., Oussalah, M. D., and Seriai, A. 2008. Evolution Styles to the Rescue of Architectural Evolution Knowledge. In Proceedings of the Third Workshop on Sharing and Reusing architectural Knowledge Architecture, Rationale, and Design Intent.

[13] Kruchten, P., Capilla, R., and Dueas, J., 2009. The Decision View's Role in Software Architecture Practice. IEEE Software, 26(2), 36-42.

[14] Kruchten, P. 2004. An ontology of architectural design decisions in software intensive systems. In 2nd Groningen Workshop on Software Variability, 54--61.

[15] Lehman, M.M. 1980. Programs, Life Cycles, and Laws of Software Evolution. Proc. IEEE Special Issue on Softw. Eng., 68(9), 1060 – 1076.

[16] Mittermeir, R. T., 2001. Software Evolution Let's Sharpen the Terminology before sharpening (out-of-scope) tools. International Workshop on Principles of Software Evolution

[17] Ozkaya, I., Diaz-Pace A., Gurfinkel, A., and Chaki, S., 2010. Using Architecturally Significant Requirements for Guiding System Evolution. In Proceedings of 14th European Conference on Software Maintenance and Reengineering.

[18] Parnas D.L. 1994. Software Aging. Proc. In Proceedings of the 16th International Conference on Software Engineering. (ICSE94) ICSE, 279- 287.

[19] Pei-Breivold, H., Crnkovic, I., 2009. Analysis of Software Evolvability in Quality Models, 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Software Process and Product Improvement (SPPI) Track, IEEE, Patras, Greece.

[20] Robson, C. 2002. Real World Research. 2nd ed. Blackwell Publishing.

[21] Strauss, A. and Corbin, J. 1998. Basics of qualitative research. SAGE Publications, p 292.

[22] Slyngstad, O. et al., 2008. Risks and Risk Management in Software Architecture Evolution: An Industrial Survey. In Software Engineering Conference. APSEC '08. 15th Asia-Pacific. 101-108.

[23] Sullivan, K., Griswold, W., Cai, Y., and Hallen, B.. 2001. The Structure and Value of Modularity in Software Design. In Proc. of FSE'01.

[24] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., and Wesslén, A. 2000. Experimentation in Software Engineering: An Introduction. 2nd ed. Springer.

[25] Yin, R.K. 2002. Case Study Research: Design and Methods. 3rd ed. Sage Publications.